10/718,420 BU-852

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 April 2002 (11.04.2002)

PCT

(10) International Publication Number
WO 02/29624 A1

(51) International Patent Classification⁷: G06F 17/30

(21) International Application Number: PCT/AU01/01250

(22) International Filing Date: 4 October 2001 (04.10.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/237,530 4 October 2000 (04.10.2000) US

(71) Applicant (*for all designated States except US*): **BULLANT TECHNOLOGY PTY LTD** [AU/AU]; Level 5, 181 Miller Street, North Sydney, NSW 2059 (AU).

(72) Inventor; and
(75) Inventor/Applicant (*for US only*): **HUETTER, Raymond, John** [AU/AU]; 14 Probate Street, Naremburn, NSW 2065 (AU).

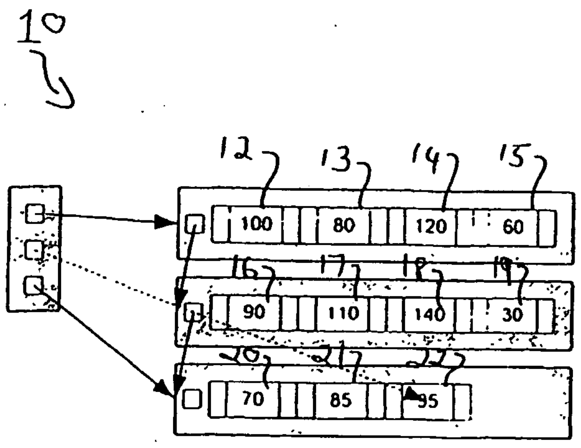
(74) Agent: **WALLINGTON-DUMMER**; Patent & Trade Mark Attorneys, P.O. Box 297, Rydalmere, NSW 1701 (AU).

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(54) Title: DATA PROCESSING STRUCTURE



Note In the above diagram the pointers between each node are not shown

(57) Abstract: A data processing structure comprising a chain of data blocks containing a binary tree structure; each of said data blocks containing a predetermined number of nodes.

WO 02/29624 A1

DATA PROCESSING STRUCTURE

The present invention relates to a data processing structure and, more particularly, to such a data structure suited to simultaneously representing work to do and work done.

BACKGROUND

In the field of data processing there are many cases where a system agent needs to iteratively walk object subgraphs and perform some processing on all the objects found within the subgraph. Starting from the root of the subgraph the agent walks the subgraph and discovers more objects which are not in the subgraph yet need to be processed.

In doing this kind of processing the agent needs some way to represent "work to do" (i.e. objects yet to be walked/processed) and be able to test if an object has been processed (i.e. objects that have been walked/processed) so as to identify and avoid cycles.

Typically a structure such as a binary tree or sparse array is used to support this kind of processing.

Often when an agent is processing a set of objects it requires a structure that enables it to record either what objects have or have not yet been processed.

However it turns out in practice that a binary tree or sparse array is rather ineffective for representing "work to do". The difference between "has/has not been processed" and "work to do" may at first appear subtle, but they are semantically different. The former is basically a testable proposition on a given handle while the later is a mutable indefinite set handles.

The performance of binary trees and sparse arrays is particularly poor when the following three things happen:

1. The structure represents work to do;
2. The structure has to be linearly scanned to discover what work to do next; and
3. While processing work further work is discovered (more objects are inserted into the structure) therefore requiring the repetition of steps 2 & 3).

The fundamental reason why steps 2 & 3 must be repeated is due to the nature/structure of the binary trees and sparse arrays. If an object appears in such a structure it always appears in a particular place.

Scanning a binary tree or sparse array can be done depth or breadth first, but there is always some traversal cursor moving through the nodes. If new values are inserted into the binary tree or sparse array (because of processing the handle already in the structure), they are inserted with indifference to the traversal cursor. The upshot is that the structure must be rescanned until there is no further work.

It is an object of the present invention to overcome or ameliorate one or more of the above mentioned disadvantages.

BRIEF DESCRIPTION OF INVENTION

Accordingly in one broad from of the invention there is provided a data processing structure comprising a chain of data blocks containing a binary tree structure; each of said data blocks containing a predetermined number of nodes.

In yet a further broad form of the invention there is provided a data processing structure comprising one or more node blocks and a control structure; said control structure grouping nodes in said node blocks according to a physical ordering in computer memory.

Preferably each said node block contains a pointer to the next node block and an array of binary tree nodes.

Preferably said binary tree node contains a value, a left pointer and a right pointer.

Preferably said control structure includes the following:

(a) the number of nodes in a block;

(b) a pointer to the first block;

(c) a pointer to the last block;

(d) a pointer to the current append block;

(e) a pointer to last node appended in the current append block;

(f) the number of free nodes in the current append block.

In yet a further broad form of the invention there is provided a method of storage of a binary tree structure, said method comprising storing nodes of said tree in node arrays; each said node array comprising a predetermined number of node storage locations; said node storage locations of said predetermined number of node storage locations having a predetermined physical ordering in memory.

In yet a further broad form of the invention there is provided a method of operation on the structure wherein a traversal cursor is utilized to keep track of work done; said cursor being advanced in a linear progression through said data processing structure as each node in said structure is processed whereby nodes through which said cursor has advanced do not require further processing.

In yet a further broad form of the invention there is provided a method of processing a directed graph wherein as further work to do is discovered in the course of progressing a traversal cursor through said graph, the work to do is inserted as a node ahead of the current location of the traversal cursor.

BRIEF DESCRIPTION OF DRAWINGS

One embodiment of the present invention will now be described with reference to the accompanying drawings wherein:

Fig. 1 represents a binary tree structure to which embodiments of the present invention can be applied;

Fig. 2 illustrates an instance of a data processing structure according to a first preferred embodiment;

Fig. 3 illustrates operations on the data processing structure of Fig. 2; and

Fig. 4 illustrates in block diagram form further detail of the data processing structure of Fig. 2.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENT

When used to represent "work to do" binary trees and sparse arrays have the following performance characteristics:

1. 0(1) test time;
2. 0(1) insert time; BUT
3. 0(N log N) … 0(N^2) scan time – depending upon usage.

With reference to Figs. 1, 2 and 3, a data processing structure 10 according to a first embodiment of the present invention comprises logically a binary tree 11 but physically held within a block chain - where each block holds a set of nodes.

With reference to Fig. 1 assume we have the logical binary tree 11 comprising a plurality of nodes 12-22 as illustrated.

Let's also assume we had added the nodes 12-22 in the following order: 12, 13, 14, 15, 16, 17, 18, 19, 20, 21 and then 22.

Again for the purposes of our discussion let us assume that our block can hold four nodes.

Therefore the structure 10 will appear as in Fig. 2.

Now if we add two new nodes 23, 24 with reference to Fig. 3, the tree structure 11 is represented as illustrated.

Observe that when a value is added into the structure 10 it is physically appended at the end of the node sequence and then the logical binary tree 11 is updated. It will be observed that structure 10 is such that the nodes have a physical ordering in memory.

By having a physical ordering in memory the structure 10 can be linearly (and efficiently) traversed and while being traversed new values can be added, such that all new values appear physically ahead of a traversal cursor. Values ahead of the traversal cursor represent work to be done, whilst values behind the traversal cursor represent work already done.

This property of adding new values ahead of the traversal cursor means the client application (not shown) can add/record further work to do. Advancing the traversal cursor eventually reaches the new work to be done.

Having multiple nodes in a block means minimal memory allocation.

The structure can be reused by resetting the pointers (thereby keeping the blocks if this is efficient), or can be deleted outright.

The structure is extensible so it can be used to hold an indefinite amount of work.

Because the structure 10 contains a binary tree 11, a value can be quickly tested to see if it is in the structure.

With particular reference to Figs. 2 and 3, the data processing structure 10 comprises:

a. One or more node blocks; and

b. A control structure.

Each node block contains:

a. A pointer to the next node block (thereby forming a chain of node blocks); and

b. An array of traditional binary tree nodes (i.e. a value, a left pointer and a right pointer).

The control structure has at least six members:

a. The number of nodes in a block;

b. A pointer to the first block;

c. A pointer to the last block;

d. A pointer to the current append block;

e. A pointer to last node appended in the current append block; and

f. The number of free nodes in the current append block.

The data processing structure 10 can allow the following operations:

1. Create - creates an entirely new data processing structure 10;

2. Destroy - completely destroys an existing data processing structure 10;

3. Add - adds an item into a data processing structure 10;

4. Test - tests if an item is in a data processing structure 10;

5. First - returns the first item in a data processing structure 10;

6. Next - returns the next item in a data processing structure 10 (given/passed the last one returned via first or next); and
7. Zero - remove all items from the data processing structure 10 and resets it but leaves the existing chain of node blocks in place for reuse.

In this embodiment the data processing structure 10 has the following properties:

1. Can be linearly scanned;
2. Further work is inserted ahead of scan;
3. 0(1) insert time;
4. 0(Log N) test time;
5. 0(N) scan time;
6. Reusable;
7. Extensible;
8. Minimal number of memory allocations;
9. High water mark of largest usage to date;
10. Easily discarded;
11. Readily tunable.

With particular reference to Fig. 4 data processing structure 10 is described in further detail and with particular reference to the method of cursor progression through the structure.

Data processing structure 10 comprises, in this instance, first node block 30, second node block 31 and third node block 32. Each of node blocks 30, 31, 32 comprises an array of four nodes 33 together with a pointer 34 to the next node block. Each node 33 includes a value 35, a left pointer 36 and a right pointer 37. In this instance each node block 30, 31, 32 comprises exactly four nodes.

A control structure 38 is associated with the data processing structure 10 and includes pointer 39 to first node block 30, pointer 40 to the current append block (in this instance node block 32), pointer 41 to last node block (in this instance third node block 32).

In addition the control structure 38 includes storage location 42 which contains the number of free nodes in the current append block (in this instance there are two spare nodes, namely spare node 42 and spare node 43 in the third node block 32).

In use a cursor 44 is moved through the control structure 10 starting with the first node of first node block 30, progressing, in this instance, from left to right through all of the nodes in first node block 30, then progressing to the first node of second node block 31, progressing through all the nodes of second node block 31 (again from left to right) and then, finally, in this instance, progressing through the two nodes of third node block 32. If, in the course of progression, the cursor 34 discovers further nodes for processing such nodes are appended to the next available node in the current append block as designated by pointer 40. Because of the structure of data processing structure 10 such additional nodes will always be placed after the current position of the cursor 44. It follows that, at any given instant, all nodes through which cursor 44 has progressed have been processed. It follows also that any nodes requiring work still to be done will always be found ahead of the current cursor position. In practice the number of nodes set aside for each node block 30, 31, 32 may be different from node block to node block. However, the number of nodes in any given node block which is set up will always be known in advance of the creation of that node block and as determined by control structure 38. In one typical example the first node block could, in practice, contain 256 nodes with subsequent node blocks each containing double the number of nodes of the previous node block.

The binary tree or sparse array implemented as the data processing structure of the preferred embodiment is an extremely useful and efficient structure for representing both work done; work to do and recording further work to do while doing work.

The above describes only some embodiments of the present invention and modifications, obvious to those skilled in the art, can be made thereto without departing from the scope and spirit of the present invention.

CLAIMS

1. A data processing structure comprising a chain of data blocks containing a binary tree structure; each of said data blocks containing a predetermined number of nodes.

2. A data processing structure comprising one or more node blocks and a control structure; said control structure grouping nodes in said node blocks according to a physical ordering in computer memory.

3. The data processing structure of Claim 2 wherein each said node block contains a pointer to the next node block and an array of binary tree nodes.

4. The data processing structure of Claim 3 where each said binary tree node contains a value, a left pointer and a right pointer.

5. The data processing structure of Claim 2 wherein said control structure includes the following:

   (a) the number of nodes in a block;

   (b) a pointer to the first block;

   (c) a pointer to the last block;

   (d) a pointer to the current append block;

   (e) a pointer to last node appended in the current append block;

   (f) the number of free nodes in the current append block.

6. A method of storage of a binary tree structure, said method comprising storing nodes of said tree in node arrays; each said node array comprising a predetermined number of node storage locations; said node storage locations of said predetermined number of node storage locations having a predetermined physical ordering in memory.

7. A method of operation on the structure of any one of claims 1 to 5 wherein a traversal cursor is utilized to keep track of work done; said cursor being advanced in a linear progression through said data processing structure as each node in said structure is processed whereby nodes through which said cursor has advanced do not require further processing.

8. A method of processing a directed graph wherein as further work to do is discovered in the course of progressing a traversal cursor through said graph, the work to do is inserted as a node ahead of the current location of the traversal cursor.

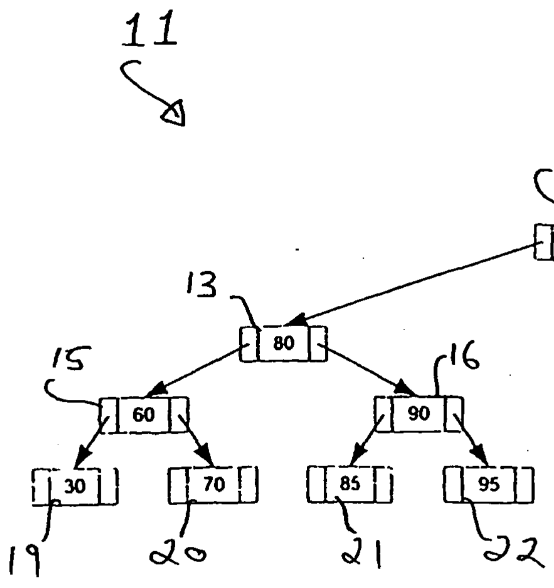
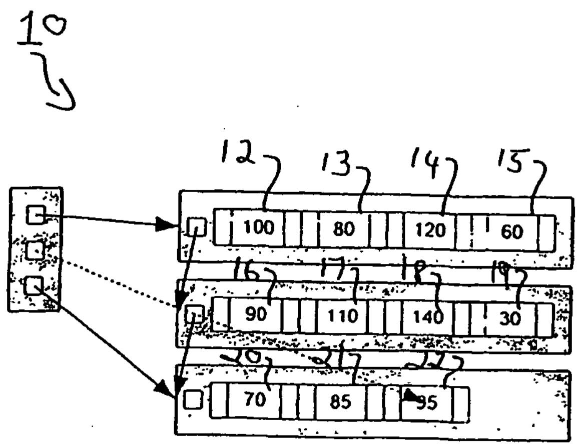
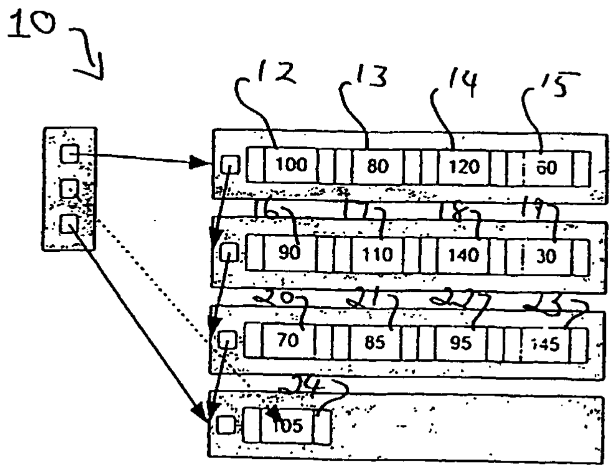

11
12
100
13
80
14
120
15
60
16
90
17
110
18
140
30
70
85
95
19
20
21
22

fig 1



Note In the above diagram the pointers between each node are not shown

fig 2



Note: In the above diagram the pointers between each node are not shown

fig 3

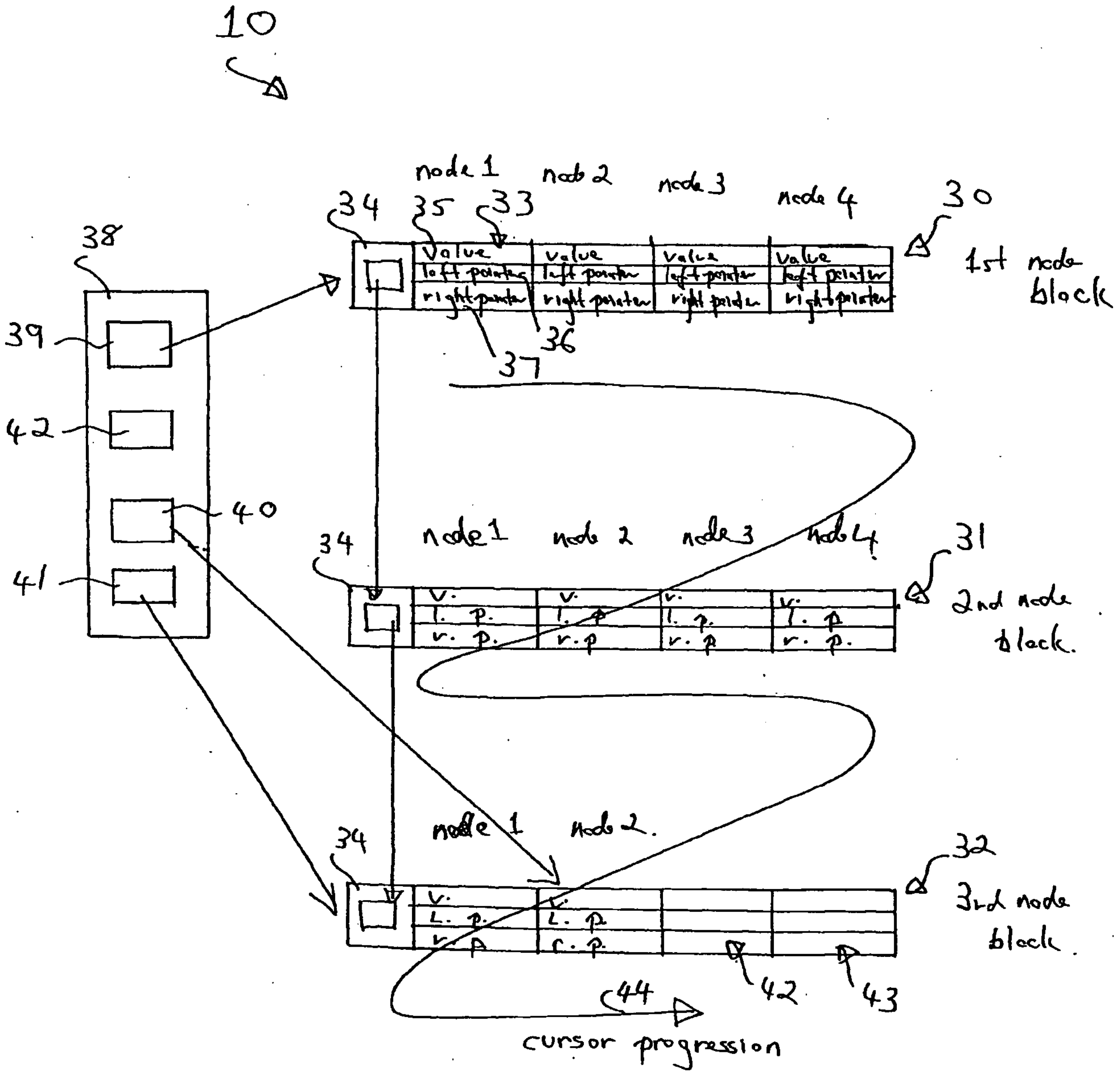

10
node 1
node 2
node 3
node 4
34
35
33
30
1st node block
38
Value
left pointer
right pointer
39
36
37
42
40
41
31
2nd node block
v.
l. p.
r. p.
32
3rd node block
43
44
cursor progression

Fig 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU01/01250

A. CLASSIFICATION OF SUBJECT MATTER

Int. Cl. ^{7}: G06F 17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
WPAT,IEEE (binary tree, blocks, nodes)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X Y	WO 98/26353 (Helix Software Co.) 18 June 1998 Abstract, page 10 lines 22-24, page 11 lines 17-21, page 12 lines 11-12, Figures 3 and 4	1,3 2,4,6
Y	US 5 644 763 (Roy) 1 July 1997 Whole document, esp. Fig. 4	1-4,6
Y	US 5 613 105 (Zbikowski et al.) 18 March 1997 column 1 and 2, Figure 8B	1

[X] Further documents are listed in the continuation of Box C [X] See patent family annex

*	Special categories of cited documents:
"A"	document defining the general state of the art which is not considered to be of particular relevance
"E"	earlier application or patent but published on or after the international filing date
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O"	document referring to an oral disclosure, use, exhibition or other means
"P"	document published prior to the international filing date but later than the priority date claimed
"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&"	document member of the same patent family

Date of the actual completion of the international search 21 November 2001	Date of mailing of the international search report 24 NOV 2001
Name and mailing address of the ISA/AU AUSTRALIAN PATENT OFFICE PO BOX 200, WODEN ACT 2606, AUSTRALIA E-mail address: pct@ipaustralia.gov.au Facsimile No. (02) 6285 3929	Authorized officer **DALE E. SIVER** Telephone No : (02) 6283 2196

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	"Implementation of an Efficient Parallel BDD Package" (Stornetta et al.) Dac'96 Whole document, esp. Sections 2, 2.1, 2.2, 3, 3.1, 3.2 and Tables	1,2
A	WO 94/02898 (Microsoft Corp.) 3 February 1994 Abstract, figures	1
A	GB 2 196 764 (Apple Computer Inc.) 5 May 1988 Abstract, figures, claims	1

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/AU01/01250

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report		Patent Family Member					
WO	98/26353	AU	58994/98	US	6208999		
US	5644763	NO	MEMBERS				
US	5613105	CA	2125606	EP	632364	JP	7028675
WO	94/02898	CA	2119788	DE	69327089	EP	606461
		JP	6511582	US	5561786		
GB	2196764	AU	80485/87	CA	1285656	DE	3736455
		FR	2606182	GB	2196764	JP	63116232
		US	4945475				

END OF ANNEX